LE RISORSE INTERNE DELLO Z80 E LA PROGRAMMAZIONE

A CURA DI

SERGIO FUMICH

Con la parola "hardware" si identifica la macchina fisica; con la parola "software" i programmi.
Un programma può essere definito come una serie di istruzioni o statements preparate in una forma comprensibile per la macchina con lo scopo di conseguire un certo risultato.
Ogni istruzione, tramite il registro decodifica delle istruzioni, è realizzata all'interno della CPU da una o più microistruzioni. Le microistruzioni, invisibili all'utente, realizzano sequenze elementari di comandi.

1. CONCETTO DI ISTRUZIONE E DI PROGRAMMA.

Con la parola "bit", abbreviazione di "binary digit", si indica l'unità elementare di informazione memorizzabile. Essa può assumere solamente o uno o l'altro dei due valori "0" (zero) e "1" (uno).
Un'informazione è generalmente rappresentata da una serie di bit. Ad esempio:

1001

è la rappresentazione del numero decimale 9 in codice binario.
Un "byte" è un insieme di bit adiacenti. Normalmente, e così è per lo Z80, sono in numero di 8. Un byte corrisponde alla minima unità indirizzabile in memoria.
Un byte può essere una qualunque delle 256 possibili differenti combinazioni di 8 cifre binarie, ciascuna delle quali può essere 0 od 1. Per individuare facilmente i singoli bit nell'ambito di un byte, i bit sono numerati da 0 a 7:

D7	D6	D5	D4	D3	D2	D1	Dϕ

La lettera "D" (iniziale di "dato") è talvolta assente. Il bit più significativo (MSB = most significant bit) è D7. Il bit meno significativo (LSB = least significant bit) è D0.
Con il termine "word" (parola) si definisce generalmente il numero di bit contigui che sono trattati in parallelo (simultaneamente) sui bus, nei registri, oppure dall'ALU. Al concetto di word si associa spontaneamente quello di lunghezza di parola. Lo Z80 ha una lunghezza di parola di otto bit.
Il carattere (character) è un insieme di bit che, secondo un dato codice, rappresentano all'interno del sistema microcalcolatore i normali simboli numerici, alfabetici e speciali.

L'istruzione è un insieme (set) di caratteri che definiscono una operazione. E', cioè, un comando codificato in forma binaria, che l'unità di elaborazione è in grado di interpretare ed eseguire in modo spontaneo, mediante le microistruzioni residenti nella CPU.
La rappresentazione binaria di un'istruzione è detta "linguaggio macchina" o "codice macchina". Ad esempio, l'istruzione in linguaggio macchina

00111100 incrementa di 1 il contenuto dell'accumulatore dello Z80. Attenzione: il linguaggio macchina è tipico del microprocessore su cui è implementato; ad esempio, nel microprocessore 6502 il codice 00111100 non è utilizzato.

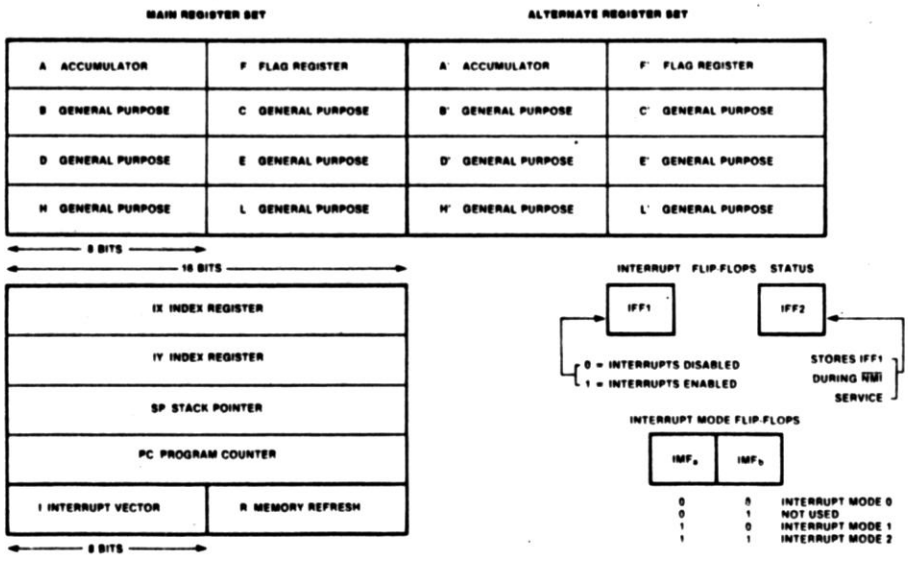
Tutti i microprocessori eseguono le istruzioni in sequenza, a meno che una delle istruzioni cambi la sequenza di esecuzione o metta l'elaboratore in "HALT". Il microprocessore, cioè, prende l'istruzione successiva del successivo indirizzo di memoria consecutivo a meno che l'istruzione corrente non gli dica specificatamente di fare altrimenti. Non è tanto la struttura circuitale hardware quanto gli algoritmi, espressi sotto forma di programma, che impongono lo svolgimento di funzioni al microprocessore.

Un programma per un elaboratore può essere definito come una sequenza di istruzioni che, prese nel loro insieme, fanno sì che l'elaboratore compia il task (compito) voluto. Un task è una operazione o un insieme di operazioni che possono essere eseguite da un elaboratore, tenuto conto della memoria e dei dispositivi di ingresso/uscita disponibili. I programmi vengono immagazzinati in memoria sotto forma di una sequenza di 0 e di 1, di bit cioè, che il microprocessore può leggere, interpretare ed eseguire in sequenza, uno alla volta.
Per lo Z80, questi bit sono memorizzati in gruppi di 8 bit, di byte cioè. Una sola istruzione può occupare uno, due, tre o quattro byte consecutivi in memoria. Lo Z80 esegue un programma leggendo un'istruzione, interpretando le configurazioni dei bit, e poi svolgendo il task necessario per completare l'operazione definita dall'istruzione. Vengono quindi lette le locazioni di memoria successive, finchè non si arriva ad un'istruzione che dice al microprocessore di fermarsi o di saltare ad un'altra locazione di memoria per eseguire l'istruzione successiva.
I programmi non sono costituiti solo da byte di istruzione. Per fornire le informazioni necessarie, nei programmi devono essere inclusi anche dei byte di dati. Ad esempio, un programma che debba sommare due numeri, deve contenere i numeri che devono essere sommati fra di loro (byte di dati) nonchè le istruzioni per eseguire l'operazione di addizione (byte di istruzione). Altri tipi di byte presenti in un programma sono i byte di indirizzo, i byte del codice di dispositivo, e i byte di displacement (spiazzamento).

Ogni microprocessore, come già accennato, ha un insieme particolare di istruzioni (instruction set) che ne riflettono la struttura interna e che dimostrano una particolare versatilità a risolvere alcuni problemi piuttosto che altri.
Il set di istruzioni di un microprocessore può essere limitato, esteso o modificabile. In particolare sono determinanti, nell'impiego di un microprocessore, le modalità di indirizzamento disponibili, le quali possono condizionare la formulazione di un algoritmo operativo in un modo piuttosto che in un altro.
Il set di istruzioni della CPU Z80, costruita dalla Zilog Corporation e prodotta anche dalla SGS-ATES, dalla Mostek e dalla SHARP (Tokio), è costituito da 158 diverse istruzioni in linguaggio macchina. Le istruzioni

dello Z80 dispongono in totale di dieci metodi di indirizzamento, con alcune istruzioni che usano due metodi contemporaneamente per accedere ai dati su cui operare.

1.1 - I registri dello Z80



Il Contatore di Programma (Program Counter) contiene l'indirizzo dell'operazione successiva che deve essere eseguita. Un semplice incrementatore connesso al contatore di programma fornisce il meccanismo automatico per prelevare le istruzioni in sequenza dalla memoria.

Lo Z80 possiede un set duplicato di registri di uso generale. Due istruzioni EXCHANGE selezionano e de-selezionano tutti i registri alternativi:

EX AF,AF'

AF ⟷ AF'

EXX

BC ⟷ BC'

DE ⟷ DE'

HL ⟷ HL'

Una volta effettuata la scelta, tutte le operazioni successive sui registri sono effettuate sull'insieme attivo finchè il prossimo scambio non scelga l'insieme non attivo.

80 pin configuration

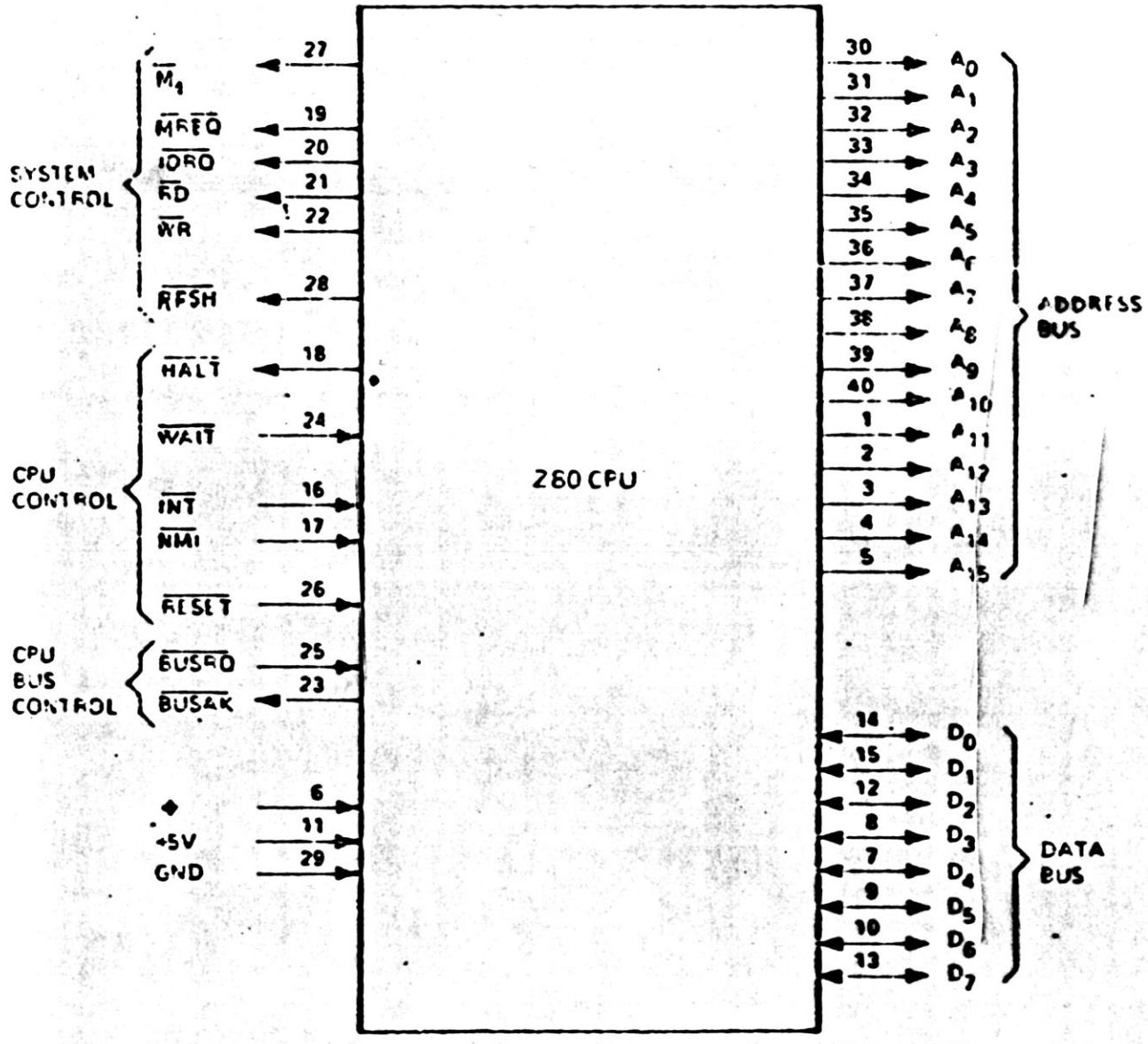


Fig 4 0-0 - CPU basic timing example

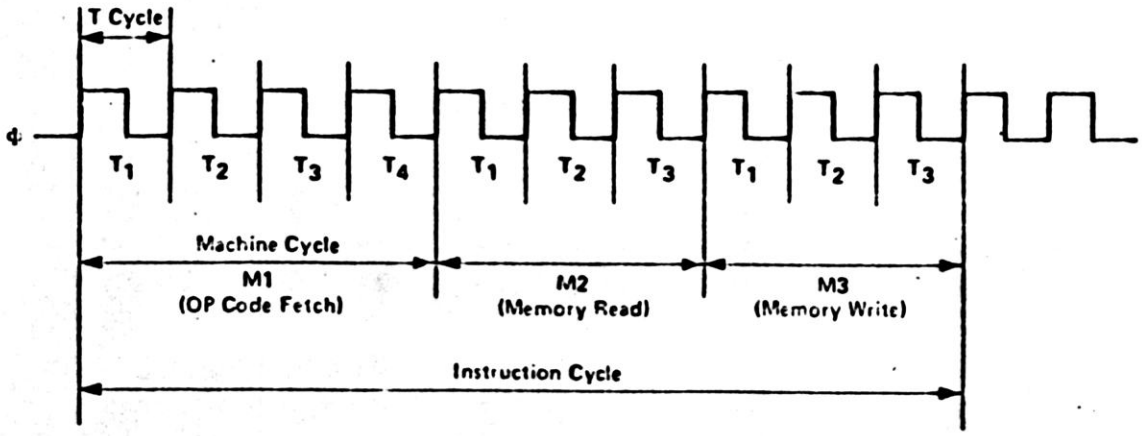
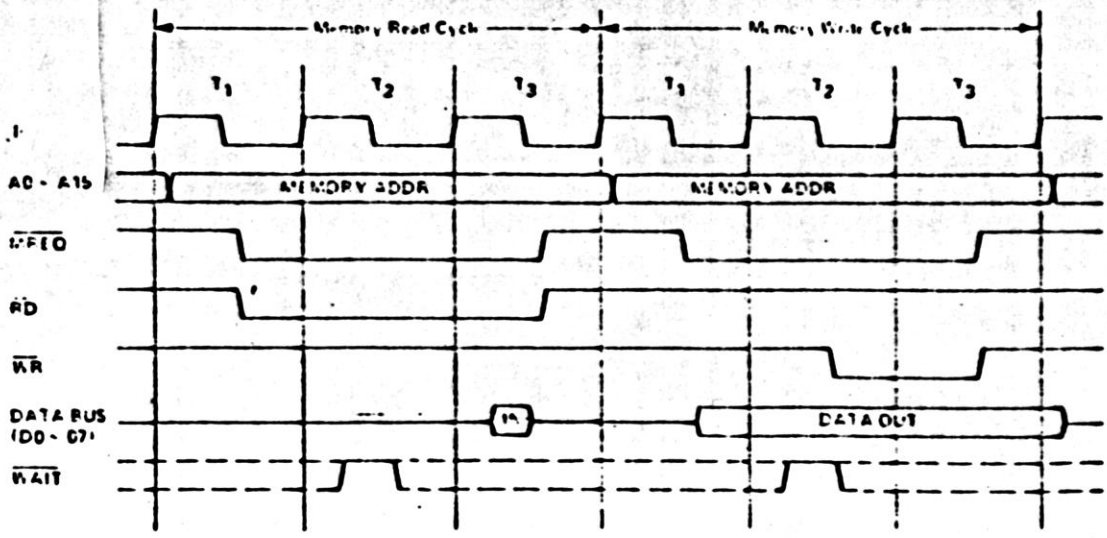


Fig. 4 0-2 - Memory read or write cycles

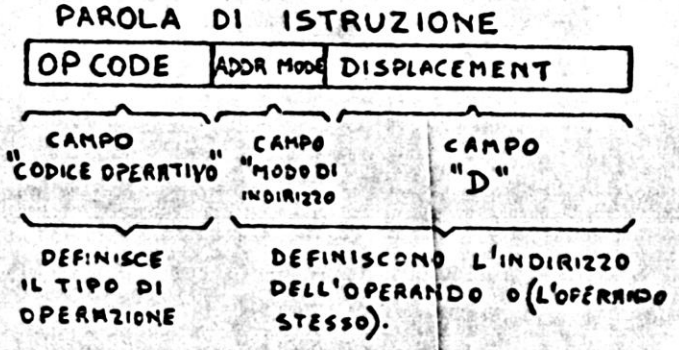


STRUTTURA DELLE ISTRUZIONI

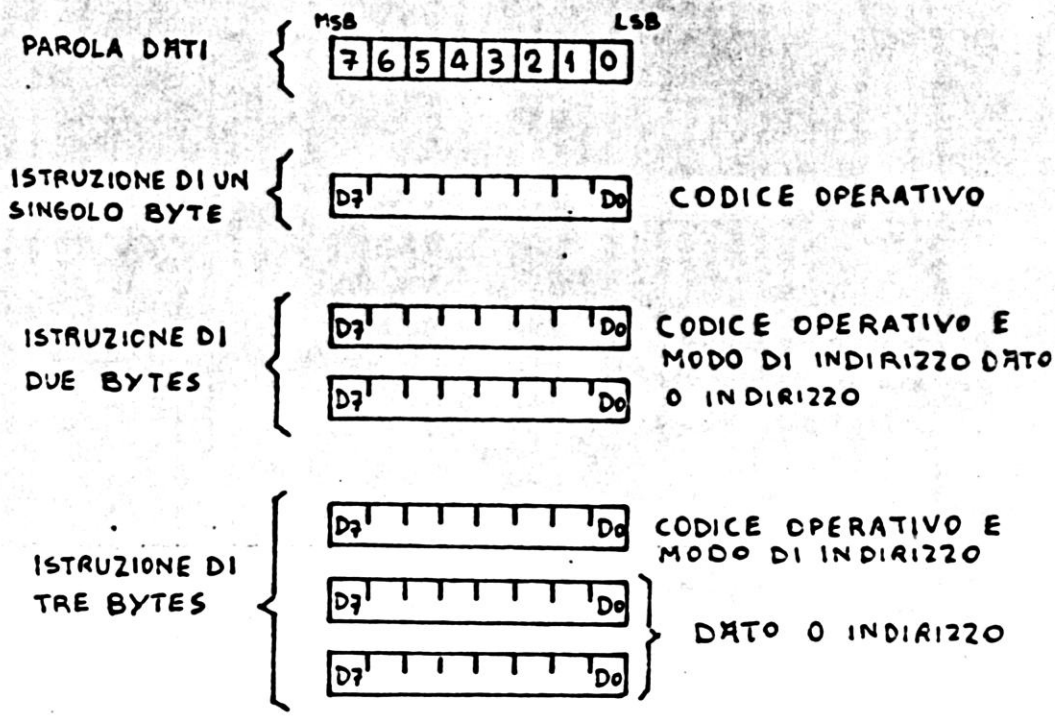
ALL'INTERNO DI OGNI ISTRUZIONE E' SPECIFICATO :

- IL TIPO DI OPERAZIONE
- IL MODO DI INDIRIZZAMENTO

LE ISTRUZIONI SONO DIVISE IN CAMPI (FIELDS)



ESEMPIO DI PAROLA-DATI E DI FORMATO ISTRUZIONI PER MICROPROCESSOR DA 8 BITS.



2. MODI DI INDIRIZZAMENTO.

2.1 INDIRIZZI DI MEMORIA.

Un indirizzo di memoria è definito come la locazione in cui viene immagazzinata una parola di memoria.
Lo Z80 può indirizzare fino a 65.536 diverse locazioni di memoria, ciascuna delle quali contiene otto bit. Questo perchè il chip ha la parola di indirizzo di 16 bit.
Lo Z80 tratta gli indirizzi di memoria a sedici bit come due byte d'indirizzo di otto bit, un byte di otto bit HI (o H) e un byte di otto bit LO (o L). Byte d'indirizzo HI (abbrevazione di high, alto) sono gli otto bit più significativi dei sedici della parola d'indirizzo del microprocessore Z80, cioè i primi 8 da sinistra. Byte d'indirizzo LO (abbreviazione di low, basso) sono gli otto bit meno significativi dei sedici bit della parola d'indirizzo del microprocessore Z80, cioè gli ultimi 8 da sinistra.

2.2 MODI DI INDIRIZZAMENTO DELLO Z80.

Quasi tutte le istruzioni dello Z80 implicano operazioni su dati che possono essere memorizzati in registri interni alla CPU, in memoria, o che possono essere trasferiti da porte di I/O. Con il termine "metodo d'indirizzamento" s'intende il metodo per mezzo del quale l'istruzione accede a questi dati.
La varietà e l'efficacia dei metodi di indirizzamento dello Z80 contribuiscono in larga misura ad avvantaggiare questo microprocessore a 8 bit rispetto ad altri, come ad esempio l'INTEL 8080A.

2.21 INDIRIZZAMENTO TRA REGISTRI.

L'indirizzamento tra registri si ha quando il codice operativo di un'istruzione contiene informazioni che specificano quali registri sono coinvolti nell'esecuzione dell'istruzione. Ad esempio, nell'istruzione LD A,B il cui codice operativo è

```
01111000
 ___ ___
   A   B
```

o in esadecimale 78, l'indirizzamento tra registri viene usato due volte, per il registro A e per il registro B.

2.22 INDIRIZZAMENTO IMMEDIATO.

Il metodo d'indirizzamento immediato si usa con le istruzioni a più byte che contengono il byte di dati a otto bit su cui bisogna operare. L'istruzione di caricamento:

```
LD  C,Ø3H
```

usa l'indirizzamento immediato. In esadecimale essa si scrive ØE Ø3. Una volta eseguita il registro C della CPU conterrà il valore esadecimale Ø3.

2.23 INDIRIZZAMENTO IMMEDIATO ESTESO.

Questo metodo d'indirizzamento richiede che l'istruzione fornisca due byte di dati in modo immediato dopo il codice operativo. Il codice macchina per qualunque istruzione che utilizzi questo metodo di indirizzamento è lungo almeno tre byte, con un byte per il codice operativo e due byte per i dati.

L'istruzione

LD BC,Ø421H

il cui codice esadecimale è Ø1 21 Ø4, usa l'indirizzamento immediato esteso. Attenzione all'ordine dei byte di dati: il byte per il registro C, 21, è il byte LO e pertanto viene per primo. Questo vale per tutti i caricamenti delle coppie di registri!

2.24 INDIRIZZAMENTO INDIRETTO TRAMITE REGISTRI.

Il metodo d'indirizzamento indiretto tramite registri fa uso di una coppia di registri per indicare dove risiedono i dati in memoria. Cioè, la coppia di registri contiene l'indirizzo del dato (byte) che l'istruzione richiede.

Una istruzione che fa uso di questo metodo è:

LD A,(HL)

il cui codice esadecimale è 7E.

Per mettere in evidenza che il contenuto della coppia di registri HL deve essere usato come puntatore di memoria, HL è racchiuso fra parentesi. E' una regola standard per l'indirizzamento indiretto.

Per alcune istruzioni, l'indirizzamento indiretto è utilizzato per accedere a due byte di dato. Il contenuto della coppia di registri specifica il byte LO e il contenuto della coppia di registri più uno punta al byte HI. Così l'istruzione:

POP BC

in esadecimale C1, carica (SP) in C e (SP+1) in B.

2.25 INDIRIZZAMENTO ESTESO.

Una istruzione che usa l'indirizzamento esteso contiene nei suoi ultimi due byte un indirizzo a 16 bit. Questo indirizzo può essere usato come puntatore della locazione di memoria per i dati richiesti, oppure può essere l'indirizzo al quale il programma dovrebbe saltare.

L'istruzione:

LD (12Ø3H),A

in esadecimale 32 Ø3 12, fa sì che nella locazione di memoria 12Ø3 (esadecimale) venga memorizzato il contenuto dell'accumulatore.

La notazione mnemonica di istruzioni di tale tipo (in cui vi è trasferimento di dati) prescrive che l'indirizzo sia racchiuso tra parentesi.

Un'istruzione come:

JP 12Ø3H

in esadecimale C3 Ø3 12, provoca, invece, il trasferimento del controllo di programma all'istruzione di indirizzo 12Ø3.

2.26 INDIRIZZAMENTO MODIFICATO IN PAGINA ZERO.

Vi sono otto istruzioni dello Z80 che utilizzano questo tipo di indirizzamento. Queste istruzioni, dette istruzioni di RESTART (riparti), fanno sì che il controllo di programma venga trasferito a parti del programma chiamate subroutine.

Tutte le istruzioni di RESTART sono lunghe un byte. Il codice operativo specifica uno degli otto possibili indirizzi (0000H, 0008H, 0010H, 0018H, 0020H, 0028H, 0030H, 0038H), uno per ogni istruzione di RESTART. Poichè il byte d'indirizzo di ordine elevato è sempre 00, il metodo d'indirizzamento è chiamato, per questo motivo, "modificato in pagina zero".

Lo scopo principale delle istruzioni di RESTART è quello di accedere a subroutine che vengono usate spesso. I vantaggi di queste istruzioni consistono nel fatto che risparmiano tempo e spazio e possono essere forzate nel microprocessore durante un'interruzione. Le istruzioni analoghe di chiamata di subroutine (CALL) usano tre byte invece dell'uno richiesto da un RESTART.

Ad esempio, l'istruzione

RST 1ØH

in esadecimale D7, trasferisce il controllo di programma alla subroutine situata a 0010H.

2.27 INDIRIZZAMENTO IMPLICITO.

Alcune istruzioni dello Z80 fanno riferimento automaticamente ad un particolare registro. Questo tipo di istruzione utilizza l'indirizzamento implicito. Il gruppo di istruzioni aritmetiche e logiche ad otto bit sono esempi di istruzioni a indirizzamento implicito, perchè implicano tutte operazioni sul contenuto dell'accumulatore.

L'istruzione:

ADD A,B

in esadecimale 8Ø, somma il contenuto del registro B all'accumulatore e carica il risultato nell'accumulatore stesso.

2.28 INDIRIZZAMENTO DI UN SINGOLO BIT.

Il set d'istruzioni dello Z80 contiene molte istruzioni che indirizzano dei singoli bit all'interno di byte immagazzinati in memoria o nei registri. Queste istruzioni che manipolano i bit usano una combinazione di metodi d'indirizzamento. L'indirizzamento tra registri, indiretto tramite registri o indicizzato, specifica la locazione di memoria o il registro della CPU che contiene il byte cui ci si vuole riferire; un codice a tre bit interno al codice operativo dell'istruzione specifica poi il bit: bit 0, 1, 2, 3, 4, 5, 6, 7. I bit sono numerati all'interno del byte a partire da destra (il bit 7 è il MSB, il bit 0 è il LSB).

L'istruzione:

SET 3,B

in esadecimale CB DØ, porta a livello logico 1 il bit numero 3 del registro B.

2.29 INDIRIZZAMENTO INDICIZZATO.

Lo Z80 ha due registri specializzati a 16 bit detti registri indice, IX e IY. Essi vengono usati principalmente con il metodo di indirizzamento indicizzato.

Con tale metodo l'indirizzo è ottenuto per somma del contenuto del registro indice e del displacement, cioè del valore relativo contenuto nel primo byte dell'istruzione dopo il codice operativo.

Ad esempio, l'istruzione

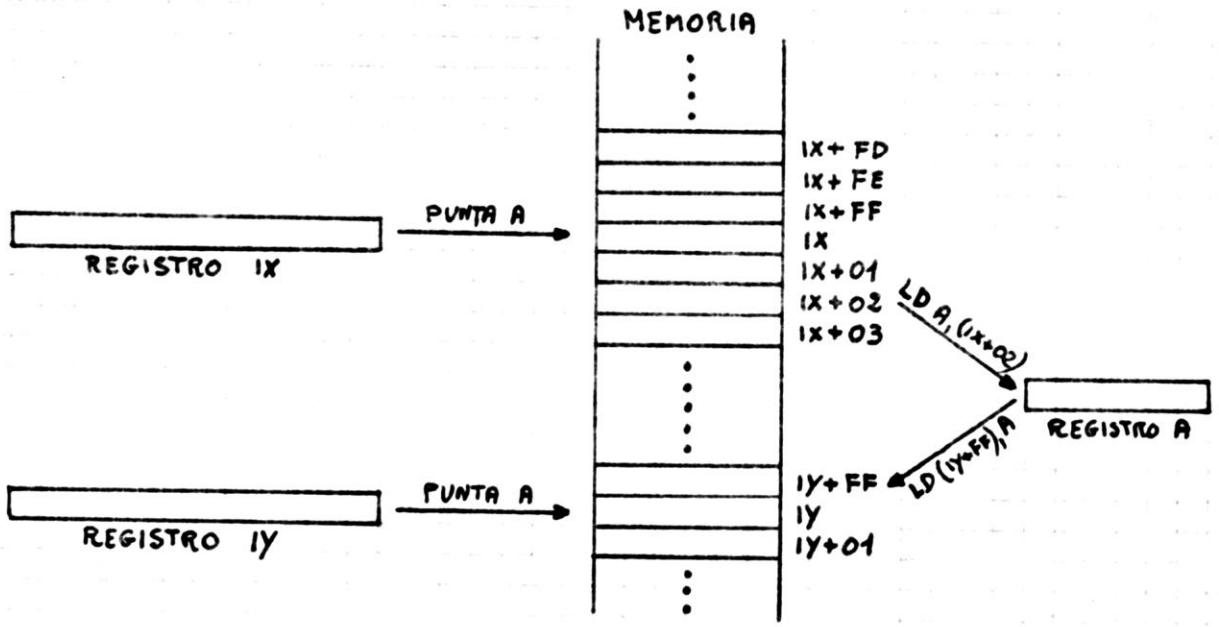
LD A,(IX+Ø2H)

in esadecimale DD 7E Ø2, carica l'accumulatore con il contenuto della locazione di memoria di due posizioni più alta rispetto alla locazione indirizzata da IX.

L'istruzione

LD (IY+FFH),A

in esadecimale FD 77 FF, memorizza il contenuto dell'accumulatore nella locazione di una posizione più bassa rispetto all'indirizzo nel registro IY (FF è il complemento a due del valore decimale -1).



L'indirizzamento indicizzato è uno strumento molto potente per accedere a tabelle di dati in memoria.

2.210 INDIRIZZAMENTO RELATIVO.

L'indirizzamento relativo è un metodo di indirizzamento molto specializzato applicabile solo a quelle istruzioni di salto dette di salto relativo (JR). Come nel caso dell'indirizzamento indicizzato il primo byte dopo il codice operativo è un numero a 8 bit complemento a due che rappresenta un displacement rispetto un dato indirizzo.

L'istruzione:

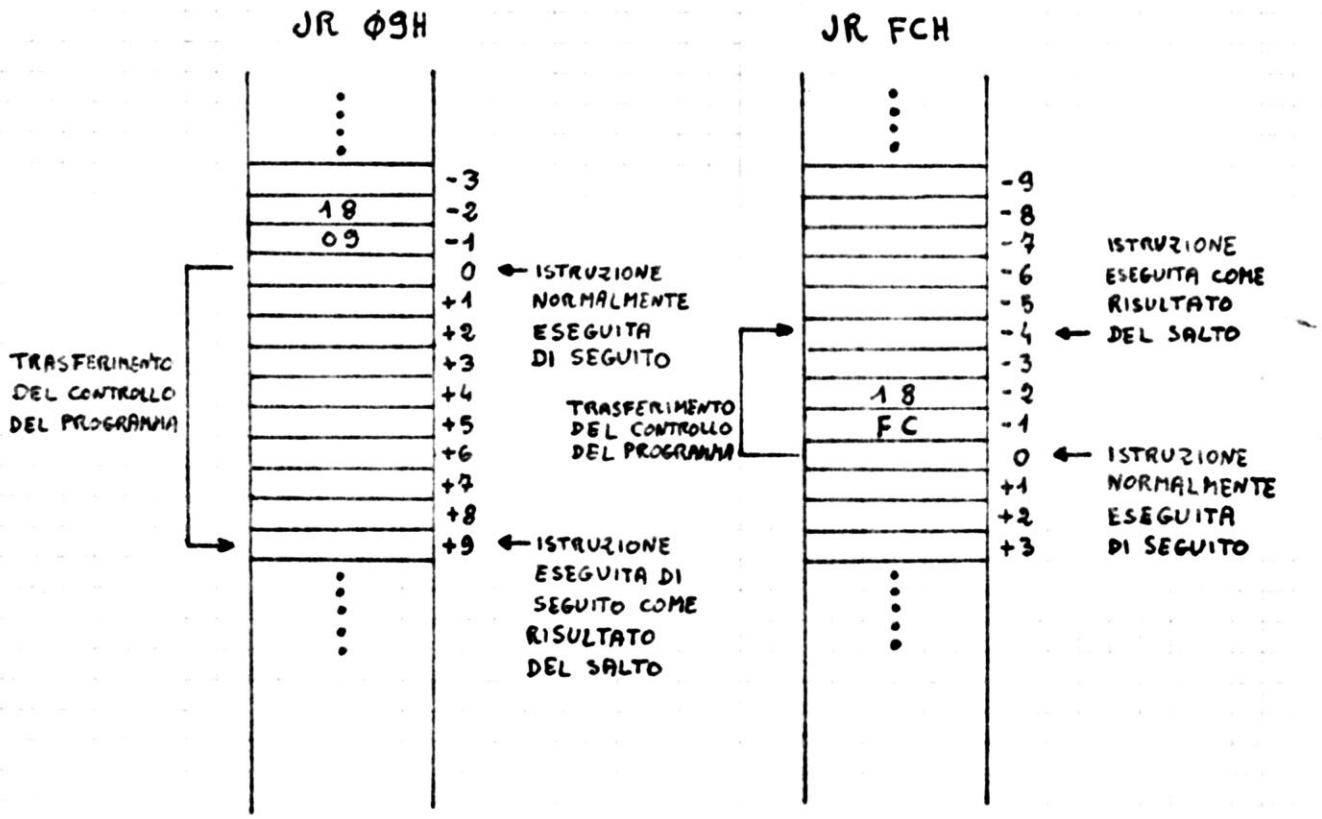
JR Ø9H

in esadecimale 18 Ø9, rappresenta un salto incondizionato relativo ad una istruzione nove byte più avanti nel programma rispetto all'istruzione che sarebbe normalmente seguita.

L'istruzione

JR FCH

in esadecimale 18 FC, trasferisce il controllo di programma indietro di quattro byte contati a partire dall'istruzione che segue l'istruzione di salto.



Il metodo d'indirizzamento relativo dello Z80 permette di scrivere codici rilocabili.

Si dice che un programma o un blocco di codici istruzione è rilocabile se è scritto in modo indipendente dalla posizione in cui risiede fisicamente in memoria. Per controllare se un programma è rilocabile basta spostare il programma, invariato, in una nuova locazione di memoria. Se il programma viene eseguito senza problemi, il programma è rilocabile. Si noti che qualunque programma con una normale istruzione di salto, che usa l'indirizzamento esteso, non è rilocabile.

Un salto normale specifica un indirizzo assoluto, per cui lo spostamento del programma in una nuova locazione richiede che questo indirizzo assoluto venga cambiato prima dell'esecuzione. La procedura di cambiamento di tutti gli indirizzi assoluti, associata al cambiamento di posizione del programma è detta rilocazione del programma.

Un altro vantaggio dei salti relativi è che richiedono solo due byte di memoria, invece dei tre byte richiesti dai salti assoluti. I salti relativi sono però limitati fra +127 e -128 locazioni.

3. TIPI DI ISTRUZIONI DELLO Z80.

Come s'è detto in precedenza, un'istruzione è un insieme di caratteri che definiscono, da soli o con altre informazioni, una operazione e che, complessivamente, fanno sì che il computer esegua l'operazione indicata. Un'operazione è definita come una specifica azione che il computer eseguirà quando un'istruzione gli dirà di farlo (per es. divisione, addizione, sottrazione, esecuzione di OR, ecc.). Il numero di operazioni differenti che un computer può eseguire e la velocità con cui le esegue sono un indice della potenza del computer stesso.

Le operazioni che lo Z80 esegue possono essere suddivise nei seguenti gruppi:

- gruppo di trasferimento dati;
- gruppo aritmetico e logico;
- gruppo di rotazione e di shift;
- gruppo di manipolazione dei bit;
- gruppo di salto, di chiamata e salto a subroutine (CALL) e di ritorno da subroutine (RETURN);
- gruppo di I/O e di controllo macchina.

Molte istruzioni appartenenti al set dello Z80 richiedono un solo byte, mentre alcune sono formate da due, tre o anche quattro byte (operazioni cosiddette multi-byte).

Il numero di byte richiesti da un'istruzione è strettamente correlato alla complessità dell'istruzione stessa e alle informazioni di cui ha bisogno.

Le istruzioni ad un solo byte richiedono solo un codice operativo e non hanno bisogno di nessuna informazione ausiliaria. La loro forma è:

codice operativo

Le istruzioni a due byte hanno quattro forme:

codice operativo/codice operativo
codice operativo/byte di dati
codice operativo/codice di dispositivo
codice operativo/byte di displacement

Le istruzioni a tre byte hanno tre forme:

codice operativo/byte di dati/byte di dati
codice operativo/byte d'indirizzo LO/byte d'indirizzo HI
codice operativo/codice operativo/byte di displacement

Le istruzioni a quattro byte hanno quattro forme:

codice operativo/codice operativo/byte di dati/byte di dati
codice operativo/codice operativo/byte d'indirizzo LO/ byte d'indirizzo HI
codice operativo/codice operativo/byte di displacement/byte di dati
codice operativo/codice operativo/byte di displacement/codice operativo

La memoria di un microcomputer Z80 è costituita da una sequenza di locazioni di memoria di 8 bit ciascuna. Quando accede alla memoria il microprocessore opera sempre trattando otto bit per volta. I tipi di informazione che possono essere immagazzinati nella memoria sono:

- codici operativi a otto bit;
- byte di dati a otto bit;
- codici di dispositivo a otto bit;

- byte d'indirizzo LO a otto bit;
- byte d'indirizzo HI a otto bit;
- byte di displacement a otto bit.

Il microprocessore distingue un tipo d'informazione dall'altro secondo l'ordine con cui appaiono le informazioni stesse. Un programma in un microcomputer inizia a funzionare ad un dato indirizzo di memoria e procede, una operazione dopo l'altra, fino all'indirizzo di memoria finale. I codici operativi nel programma dicono cosa aspettarsi, cioè se il byte successivo è un byte di dati, di indirizzo, di codice di dispositivo o un altro codice operativo.

4. CONSIDERAZIONI GENERALI SUL FORMATO DELLE ISTRUZIONI E CODICI OPERATIVI.

Abbiamo visto come un'istruzione, per poter essere interpretata dal microprocessore, debba essere formulata come sequenza binaria di 0 e 1. Per sapere quanto è lunga questa sequenza binaria bisogna considerare il parallelismo del microprocessore.

Nell'ipotesi di un microprocessore a otto bit, l'istruzione sarà lunga un byte, ma non sempre sono sufficienti, come già si è osservato per lo Z80, otto bit per descrivere tutti i tipi di istruzione e i modi di indirizzamento, in particolare tutte le combinazioni fra le istruzioni e gli indirizzamenti possibili.

Concettualmente un'istruzione è divisa in un certo numero di bit che formano il codice operativo (OP CODE), e da alcuni bit che formano il campo indirizzo. Il numero di bit necessari per descrivere il codice operativo è funzione del numero di istruzioni presenti nel microprocessore. Con 6 bit ad esempio possono essere descritte 64 istruzioni diverse ($2^6 = 64$).

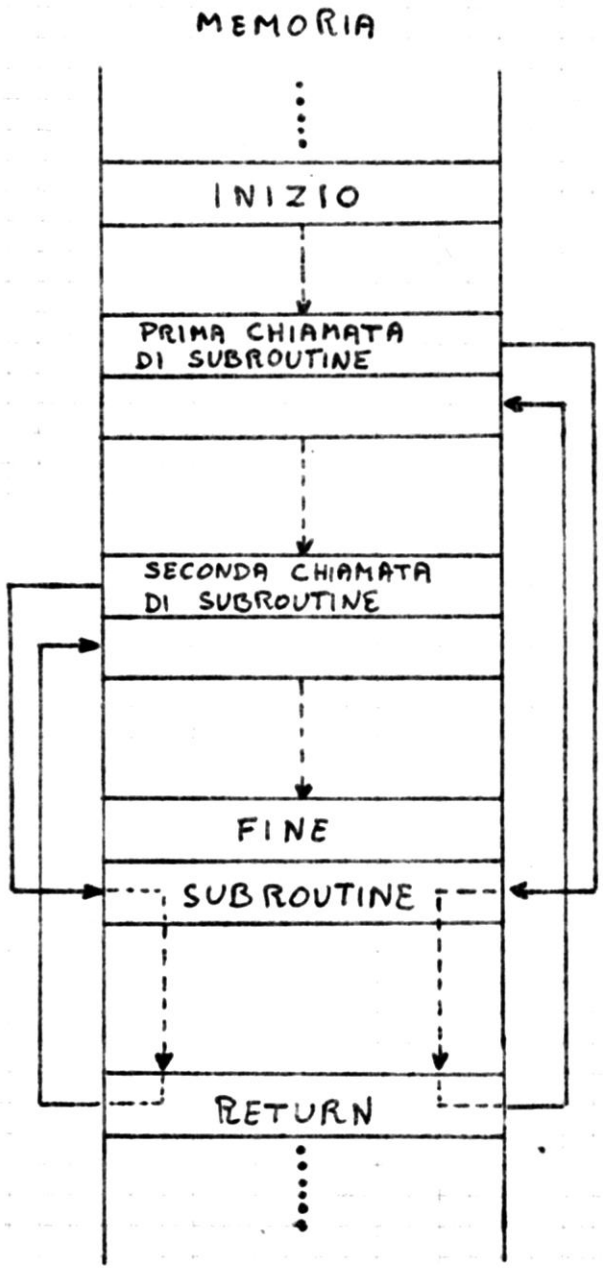
Il campo indirizzo contiene tutte le informazioni circa le modalità di indirizzamento vero e proprio. La lunghezza di questo campo dipende dalla struttura delle microistruzioni del microprocessore in particolare dai tipi di indirizzamento possibili. Se fosse, ad esempio, previsto un indirizzamento diretto nell'ambito di una pagina da 4K (4096 celle di memoria), il campo indirizzi dovrebbe essere lungo 12 bit ($2^{12} = 4K$) e il campo codice operativo almeno 6 bit. Un'istruzione di questo tipo richiede 18 bit, sarà dunque lunga 3 byte.

Se invece fosse possibile indirizzare direttamente solo una pagina da 1K (1024 celle di memoria) il campo indirizzi scenderebbe a 10 bit ($2^{10} = 1024 = 1K$), e l'istruzione in totale richiederebbe 16 bit, due byte, con un risparmio notevole rispetto al primo. Con un'istruzione lunga più byte sono necessarie più fasi di fetch e quindi un tempo maggiore di esecuzione. Una parola corta vuol dire basso costo del dispositivo ma istruzioni meno potenti; una parola lunga (ad esempio 16 bit) vuol dire maggiore velocità e istruzioni più potenti, cioè minor numero di istruzioni per eseguire una operazione e minore occupazione di memoria.

5. SUBROUTINE.

Il sottoprogramma o subroutine è una particolare sequenza di istruzioni che implementano una funzione più volte richiesta nell'esecuzione del

programma principale.
Non è conveniente scrivere più volte nel programma un insieme di istruzioni che si ripetono uguali a sè stesse, ma è conveniente scrivere un modulo (subroutine) una volta per tutte e richiamarlo ogni volta che sia necessario svolgere quella funzione.
La figura esemplifica la chiamata a subroutine.



Per ricordare l'indirizzo a cui deve ritornare il program counter al termine dell'esecuzione della subroutine, è necessario disporre di una memoria temporanea, la quale può essere interna o esterna. Occorre quindi memorizzare un indirizzo, normalmente 16 bit. Un solo registro a 16 bit sarebbe sufficiente, ma così si consentirebbe un solo livello di sot-

toprogramma, cioè non si potrebbero chiamare subroutine dall'interno di altre subroutine. Per avere più livelli di sottoprogrammi occorrono più registri e questo è ottenuto usando lo stack, cioè una pila di registri.
Per comprendere meglio cosa sia uno stack può forse aiutare l'analogia con il seguente esempio preso dalla vita di tutti i giorni. Siamo in un self-service. I vassoi puliti di cui i clienti si servono si trovano accatastati sul banco. Il vassoio che conviene prendere è quello che sta in cima al mucchio. Perciò, man mano che vengono serviti nuovi clienti i vassoi usati sono stati lavati e asciugati e vengono "spinti" (pushed) o infilati sopra il mucchio (stack). La relazione che regola il comportamento della pila può essere così riassunta:

LAST IN, FIRST OUT

cioè l'ultimo a entrare è il primo a uscire.
Questa regola, LIFO, è quella che governa gli stack come strutture di dati. Vediamo un esempio.
Di seguito è riportato un elenco di parte della memoria, con ogni locazione contrassegnata dal suo indirizzo. Parlando delle operazioni dello stack, gli indirizzi di memoria aumentano man mano che si scende verso il basso. Lo stack pointer, registro SP nello Z80, punta sempre al byte dello stack che ha l'indirizzo più basso.

SP	00
SP+1	01
SP+2	02
SP+3	03
	⋮

Si può pensare a questo insieme di locazioni di memoria come ad uno stack in cui la locazione SP (che sta per stack pointer) rappresenta l'indirizzo del byte in cima allo stack. Nella figura il contenuto della parte alta dello stack, (SP), è 00.
Su questo stack di byte si possono eseguire due operazioni:
- estrarre i byte dalla parte alta (POP);
- inserire nuovi byte nello stack (PUSH).

Entrambe le operazioni danno origine ad un nuovo byte in cima allo stack.

Il microprocessore Z80 ha due istruzioni, POP e PUSH, che eseguono le suddette operazioni per due byte alla volta. Entrambe le istruzioni richiedono che una coppia di registri sia specificata come sorgente (per PUSH) o come destinazione (per POP) dei due byte di dati che vengono trasferiti.

Si fa riferimento allo stack quando si vogliono mettere da parte sia per comodità di programmazione, sia per carenza di accumulatori disponibili, e soprattutto per salvare il valore attuale del program counter (PC) quando si vuole abbandonare la sequenza di programma in atto per eseguire

una chiamata a subroutine. Lo stack permette quindi di salvare il contenuto del PC tante volte consecutive, quanti sono i suoi registri. Può così essere realizzata una struttura di chiamate a subroutine una dentro l'altra, cioè un nesting (nidificazione) di sottoprogrammi.

Lo stack, a seconda del tipo di microprocessore, può essere interno ad esso o esterno. In questo secondo caso lo stack è realizzato in una zona di memoria RAM ed all'interno della CPU esiste un registro dedicato, lo stack pointer, che viene incrementato e decrementato automaticamente.

La realizzazione di uno stack esterno presenta dei vantaggi e degli svantaggi. Tra i vantaggi va ricordata la non limitazione alla nidificazione dei sottoprogrammi e la possibilità di usare lo stack come area di memoria temporanea per il passaggio, ad esempio, dei parametri ad una subroutine. Lo svantaggio risiede nella necessità di avere memoria RAM, soluzione non sempre compatibile con sistemi minimi.